

WEST[Help](#)[Logout](#)[Interrupt](#)[Main Menu](#)[Search Form](#)[Posting Counts](#)[Show \\$ Numbers](#)[Edit \\$ Numbers](#)[Preferences](#)[Cases](#)**Search Results -**

Terms	Documents
(extend\$ or expand\$) same macro near language	19

Database:

US Patents Full-Text Database
 US Pre-Grant Publication Full-Text Database
 JPO Abstracts Database
 EPO Abstracts Database
 Derwent World Patents Index
 IBM Technical Disclosure Bulletins

Search:

[Refine Search](#)[Recall Text](#)[Clear](#)**Search History**
DATE: Tuesday, November 19, 2002 [Printable Copy](#) [Create Case](#)
Set Name Query
 side by side

Hit Count Set Name
 result set

DB=USPT,PGPB,JPAB,EPAB,DWPI,TDBD; PLUR=YES; OP=OR

<u>L37</u>	(extend\$ or expand\$) same macro near language	19	<u>L37</u>
<u>L36</u>	L35 and repository	2	<u>L36</u>
<u>L35</u>	L34 and keyword	8	<u>L35</u>
<u>L34</u>	extend\$ same macro same language	45	<u>L34</u>
<u>L33</u>	L32 and extend	70	<u>L33</u>
<u>L32</u>	macro near language	345	<u>L32</u>
<u>L31</u>	extend\$ near language	333	<u>L31</u>
<u>L30</u>	extend\$ near macro near language	2	<u>L30</u>
<u>L29</u>	((((717/140)!.CCLS.))	161	<u>L29</u>
<u>L28</u>	((((717/124)!.CCLS.))	187	<u>L28</u>
<u>L27</u>	((((717/122)!.CCLS.))	72	<u>L27</u>
<u>L26</u>	((((717/118)!.CCLS.))	81	<u>L26</u>

<u>L25</u>	((717/117)!.CCLS.))	45	<u>L25</u>
<u>L24</u>	((717/116)!.CCLS.))	154	<u>L24</u>
<u>L23</u>	((717/115)!.CCLS.))	37	<u>L23</u>
<u>L22</u>	((717/114)!.CCLS.))	139	<u>L22</u>
<u>L21</u>	((717/106)!.CCLS.))	117	<u>L21</u>
<u>L20</u>	((712/209)!.CCLS.))	214	<u>L20</u>
<u>L19</u>	((345/34\$)!.CCLS.))	289	<u>L19</u>
<u>L18</u>	((345/348)!.CCLS.))	0	<u>L18</u>
<u>L17</u>	((345/\$)!.CCLS.))	48575	<u>L17</u>
<u>L16</u>	((703/\$)!.CCLS.))	5095	<u>L16</u>
<u>L15</u>	((712/\$)!.CCLS.))	8373	<u>L15</u>
<u>L14</u>	((712/3)!.CCLS.))	42	<u>L14</u>
<u>L13</u>	((703/3)!.CCLS.))	147	<u>L13</u>
<u>L12</u>	((717/8)!.CCLS.))	0	<u>L12</u>
<u>L11</u>	((717/3)!.CCLS.))	0	<u>L11</u>
<u>L10</u>	((717/\$)!.CCLS.))	4028	<u>L10</u>
<u>L9</u>	((707/\$)!.CCLS.))	17299	<u>L9</u>
<u>L8</u>	((707/100)!.CCLS.))	1289	<u>L8</u>
<u>L7</u>	((707/2)!.CCLS.))	1221	<u>L7</u>
<u>L6</u>	((707/526)!.CCLS.))	302	<u>L6</u>
<u>L5</u>	((707/513)!.CCLS.))	885	<u>L5</u>
<u>L4</u>	((707/500)!.CCLS.))	464	<u>L4</u>
<u>L3</u>	((707/8)!.CCLS.))	578	<u>L3</u>
<u>L2</u>	((707/3)!.CCLS.))	2254	<u>L2</u>
<u>L1</u>	((707/1)!.CCLS.)	1944	<u>L1</u>

END OF SEARCH HISTORY

WEST Search History

DATE: Tuesday, November 19, 2002

<u>Set Name</u> side by side	<u>Query</u>	<u>Hit Count</u>	<u>Set Name</u> result set
<i>DB=USPT; PLUR=YES; OP=OR</i>			
L68	5477451.pn.	1	L68
L67	5852740.pn.	1	L67
L66	4506326.pn.	1	L66
L65	4686623.pn.	1	L65
L64	4688195.pn.	1	L64
L63	4729096.pn.	1	L63
L62	4751635.pn.	1	L62
L61	4787035.pn.	1	L61
L60	5854750.pn.	1	L60
L59	5884078.pn.	1	L59
L58	5928360.pn.	1	L58
L57	5938766.pn.	1	L57
L56	4667290.pn.	1	L56
L55	5167023.pn.	1	L55
L54	5179703.pn.	1	L54
L53	5204960.pn.	1	L53
L52	5367683.pn.	1	L52
L51	5625822.pn.	1	L51
L50	5649203.pn.	1	L50
L49	5692196.pn.	1	L49
L48	5768593.pn.	1	L48
L47	6151703.pn.	1	L47
L46	6134581.pn.	1	L46
L45	6151703.pn.	1	L45
L44	6173441.pn.	1	L44
L43	6175954.pn.	1	L43
L42	6226675.pn.	1	L42
L41	6233586.pn.	1	L41
L40	6237135.pn.	1	L40
L39	6266666.pn.	1	L39
L38	6266716.pn.	1	L38
<i>DB=USPT,PGPB,JPAB,EPAB,DWPI,TDBD; PLUR=YES; OP=OR</i>			
L37	(extend\$ or expand\$) same macro near language	19	L37

L36	L35 and repository	2	L36
L35	L34 and keyword	8	L35
L34	extend\$ same macro same language	45	L34
L33	L32 and extend	70	L33
L32	macro near language	345	L32
L31	extend\$ near language	333	L31
L30	extend\$ near macro near language	2	L30
L29	((((717/140)!.CCLS.))	161	L29
L28	((((717/124)!.CCLS.))	187	L28
L27	((((717/122)!.CCLS.))	72	L27
L26	((((717/118)!.CCLS.))	81	L26
L25	((((717/117)!.CCLS.))	45	L25
L24	((((717/116)!.CCLS.))	154	L24
L23	((((717/115)!.CCLS.))	37	L23
L22	((((717/114)!.CCLS.))	139	L22
L21	((((717/106)!.CCLS.))	117	L21
L20	((712/209)!.CCLS.)	214	L20
L19	((345/34\$)!.CCLS.))	289	L19
L18	((345/348)!.CCLS.))	0	L18
L17	((345/\$)!.CCLS.))	48575	L17
L16	((703/\$)!.CCLS.))	5095	L16
L15	((712/\$)!.CCLS.))	8373	L15
L14	((712/3)!.CCLS.))	42	L14
L13	((703/3)!.CCLS.))	147	L13
L12	((717/8)!.CCLS.))	0	L12
L11	((717/3)!.CCLS.))	0	L11
L10	((717/\$)!.CCLS.))	4028	L10
L9	((707/\$)!.CCLS.))	17299	L9
L8	((707/100)!.CCLS.))	1289	L8
L7	((707/2)!.CCLS.))	1221	L7
L6	((707/526)!.CCLS.))	302	L6
L5	((707/513)!.CCLS.))	885	L5
L4	((707/500)!.CCLS.))	464	L4
L3	((707/8)!.CCLS.))	578	L3
L2	((707/3)!.CCLS.))	2254	L2
L1	((707/1)!.CCLS.)	1944	L1

END OF SEARCH HISTORY

Colbert, Ella

From: richard.ellis@uspto.gov
Sent: Tuesday, November 19, 2002 11:00 AM
To: leigh.garbowski@uspto.gov; paul.myers@uspto.gov; stephen.meier@uspto.gov; ella.colbert@uspto.gov; joseph.valenza@uspto.gov; Michael.Shingleton@uspto.gov; david.robertson@uspto.gov; jacques.louis-jacques@uspto.gov; stacy.whitmore@uspto.gov; samuel.broda@uspto.gov; andrew.caldwell@uspto.gov
Subject: Printer Reminder

This is an automated reminder to remember to read your designated group printers.

DEFINING AND USING MACROS

The Lynx language of LinkWinds can be used to create **Macros**, a series of commands that may be re-executed at any time during a session. **Macros** are created through the top-level "**Macros**" menu button. To start creating a macro, select "Start Macro". All subsequent LinkWinds operations will be saved in a file with the .lynx extension and a name dependent upon the date and time. The recording ends when "End Macro" is selected.

The **Macro** created is saved in the directory from which LinkWinds was executed. There are two different save modes selectable from the menu. Selecting "Temporary" allows the user to execute the macro during the current session only. The file itself is still saved, but the user will only have access to it in future sessions if an entry for it is manually edited into the lw.macros file. If the save mode is "Permanent", then the macro is automatically recorded in the lw.macros file and will be available as a menu item in all subsequent LinkWinds sessions unless removed. The name given is the same as the macro file name, minus the extension. The user may of course change the name by editing the lw.macros file.

The file lw.macros contains a listing of all **Macros** to which the user will have access through the top-level "**Macros**" menu. Each macro requires two entries. The first is the name of the macro as it will appear in the "**Macros**" menu. Ideally, this should be descriptive. The second entry is the actual name of the file containing the macro. This file must exist somewhere in the paths given in lw.config. As with other LinkWinds text files, anything following a # sign on a line is treated as a comment and ignored. Also note that the name for the file containing the listing of **Macros** is specified in lw.config and can be changed if the user desires.

Two sample **Macros** are shipped with LinkWinds. The first is "Startup" which initializes a sample session by placing several objects on the screen and altering the states of some of these objects. It is an example of how **Macros** can be used to configure an initial LinkWinds setup so that the user is not required to repeat the same steps at the beginning of each session. The second sample is Expand, which simply stretches the image displayed in the Image1 application.

WEST

Generate Collection

Print

L1: Entry 2 of 5

File: USPT

Feb 27, 1996

DOCUMENT-IDENTIFIER: US 5495613 A

TITLE: Method and apparatus for extending the capability of a system editor using high-level language transforms

Abstract Text (1):

The present invention is directed to a technique extending the commands, and consequently, the capability of a system editor. Rather than extending the editor with existing editor commands, as in the use of macros, the present invention enables the user to extend the editor using new commands, called transforms. The transforms are written in a standard high level language such as ALGOL, C, COBOL, FORTRAN, or PASCAL. A transform is a new type of command that may be added to a system editor. The transform command is created, compiled, and then stored in a library. When the command is called, an interface in the system editor provides access to the transforms in the library. Transforms are designed so that, to the user, they operate like normal editor commands.

Drawing Description Text (8):

FIG. 3a is a screen display of exemplary commands such as macros and transforms used to extend the system editor.

Detailed Description Text (3):

The present invention is directed to extending the commands and, consequently, the capability of a system editor. Rather than extending the capabilities of a system editor with existing editor commands, as in the use of macros, the present invention enables the user to extend the editor using new commands, called transforms. The transforms are written in a standard high level language. A high level language is a programming language that does not reflect the structure of any one given computer or that of any given class of computers, such as ALGOL, C, COBOL, FORTRAN, or PASCAL. This facilitates translation of a computer program written in the high level language into several different machine codes.

Detailed Description Text (14):

In addition to using the core commands, a user of the system editor may create macros. FIG. 3a is a screen display of exemplary commands such as macros and transforms used to extend the system editor. Macros are constructed of existing commands and are used to execute a sequence of core commands, macros and/or transforms that are used frequently in combination with one another.

Detailed Description Text (17):

Transforms, like macros, extend the capability of system editors by supplementing available commands.

WEST

Generate Collection

Print

L1: Entry 2 of 5

File: USPT

Feb 27, 1996

US-PAT-NO: 5495613

DOCUMENT-IDENTIFIER: US 5495613 A

TITLE: Method and apparatus for extending the capability of a system editor using high-level language transforms

DATE-ISSUED: February 27, 1996

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Brody; Ronald E.	Exton	PA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Unisys Corporation	Blue Bell	PA			02

APPL-NO: 08/ 292613 [PALM]

DATE FILED: August 18, 1994

INT-CL: [06] G06 F 3/00

US-CL-ISSUED: 395/700; 364/280.7, 364/226.6, 364/DIG.1

US-CL-CURRENT: 717/110; 715/530, 717/114

FIELD-OF-SEARCH: 395/600, 395/700, 364/419

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected

Search ALL

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
<input type="checkbox"/>	4827410	May 1989	Corren	395/155
<input type="checkbox"/>	4908612	March 1990	Bromley et al.	340/706
<input type="checkbox"/>	5043891	August 1991	Goldstein et al.	364/419

OTHER PUBLICATIONS

Alan Simpson, "Mastering WordPerfect 5.1 & 5.2 for Windows" Sybex Inc., pp. 511-521, 874-906, 1993.

ART-UNIT: 237

PRIMARY-EXAMINER: Black; Thomas G.

ASSISTANT-EXAMINER: Wang; Peter Y.

ATTY-AGENT-FIRM: Ratner & Prestia

ABSTRACT:

The present invention is directed to a technique extending the commands, and

consequently, the capability of a system editor. Rather than extending the editor with existing editor commands, as in the use of macros, the present invention enables the user to extend the editor using new commands, called transforms. The transforms are written in a standard high level language such as ALGOL, C, COBOL, FORTRAN, or PASCAL. A transform is a new type of command that may be added to a system editor. The transform command is created, compiled, and then stored in a library. When the command is called, an interface in the system editor provides access to the transforms in the library. Transforms are designed so that, to the user, they operate like normal editor commands.

12 Claims, 20 Drawing figures

WEST



Generate Collection

Print

L1: Entry 3 of 5

File: JPAB

Sep 11, 1992

PUB-NO: JP404257034A

DOCUMENT-IDENTIFIER: JP 04257034 A

TITLE: LOGICAL NAME DISPLAY SYSTEM FOR MACRO FILE

PUBN-DATE: September 11, 1992

INVENTOR-INFORMATION:

NAME

COUNTRY

OKURA, TAKAO

ASSIGNEE-INFORMATION:

NAME

COUNTRY

MITSUBISHI ELECTRIC CORP

APPL-NO: JP03039595

APPL-DATE: February 8, 1991

INT-CL (IPC): G06F 11/28; G06F 12/00

ABSTRACT:

PURPOSE: To open a macro file by a command used for the debug of a macro or the like, and to take out a logical file name.

CONSTITUTION: After the logical file name including an extender is set in the first records R1 of macro files 2a-2n under a prescribed directory, a system program 1 transmits a command MDIR at the time of opening the file, searches the macro file to be opened by each extender, and displays the logical file name at a CRT 3 after opening the searched file.

COPYRIGHT: (C)1992,JPO&Japio

WEST

Generate Collection

Print

L1: Entry 4 of 5

File: DWPI

Jul 31, 2002

DERWENT-ACC-NO: 2000-364943

DERWENT-WEEK: 200279

COPYRIGHT 2003 DERWENT INFORMATION LTD

TITLE: Extensible macro language providing method for use in computer language processors, involves retrieving code associated with keywords representing new macro command, which is then executed

Standard Title Terms (1):

EXTEND MACRO LANGUAGE METHOD COMPUTER LANGUAGE PROCESSOR RETRIEVAL CODE ASSOCIATE
KEYWORD REPRESENT NEW MACRO COMMAND EXECUTE

WEST**End of Result Set**

Generate Collection

Print

L1: Entry 5 of 5

File: DWPI

Jul 30, 1986

DERWENT-ACC-NO: 1987-085544

DERWENT-WEEK: 198712

COPYRIGHT 2003 DERWENT INFORMATION LTD

TITLE: Microcommands memory two-level control - has extended logic handling of micro command details offering scope for modification to whole or parts

Basic Abstract Text (1):

Calculating system element for microcommand processing is intended for wider applicability by broadening the microcommand basis. The first, setting-up mode is used in extending the system macro-functions, e.g. command list, esp. when working with new operations systems. Each new micro-program enters setting-up memory (3) via input (19) and address counter (8). The next, main, mode consists of entering the operation code determining the initial microprogram address in counter (8) and register (7). Synchro pulse (20) moves it through OR-gates (13) to address register (4) and, delayed by delay (15), this pulse retrieves the address microcommand from memory (1). Three different fields here determine the function flow: field (1.1) for successive microcommand addresses circulates signals back via OR-gates (13) without modification: field (1.2) holds the code of the logic condition(s) being tested, which loops back to module-2 adders (12) for combination with register (6) containing transfer details to the next linear portion of the microprogram. Field (1.3) holds codes of the addresses of the zones of operations microcommands, for transfer through second address register (5); zone information is extracted by control signal (22) from elements (2) to demultiplexers (11), OR-gates (14), microcommand output register (10).



WEST

Generate Collection

Print

Search Results - Record(s) 1 through 5 of 5 returned.☐ 1. Document ID: US 5712990 A

L1: Entry 1 of 5

File: USPT

Jan 27, 1998

US-PAT-NO: 5712990

DOCUMENT-IDENTIFIER: US 5712990 A

TITLE: Economical automated process for averting physical dangers to people, wildlife or environment due to hazardous waste

DATE-ISSUED: January 27, 1998

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Henderson; Don J.	Danville	CA		

US-CL-CURRENT: 705/28; 705/29

Full	Title	Citation	Front	Review	Classification	Date	Reference	Sequences	Attachments	Claims	KWIC	Draw Desc	Image
------	-------	----------	-------	--------	----------------	------	-----------	-----------	-------------	--------	------	-----------	-------

☐ 2. Document ID: US 5495613 A

L1: Entry 2 of 5

File: USPT

Feb 27, 1996

US-PAT-NO: 5495613

DOCUMENT-IDENTIFIER: US 5495613 A

TITLE: Method and apparatus for extending the capability of a system editor using high-level language transforms

DATE-ISSUED: February 27, 1996

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Brody; Ronald E.	Exton	PA		

US-CL-CURRENT: 717/110; 715/530, 717/114

Full	Title	Citation	Front	Review	Classification	Date	Reference	Sequences	Attachments	Claims	KWIC	Draw Desc	Image
------	-------	----------	-------	--------	----------------	------	-----------	-----------	-------------	--------	------	-----------	-------

☐ 3. Document ID: JP 04257034 A

L1: Entry 3 of 5

File: JPAB

Sep 11, 1992

PUB-NO: JP404257034A

DOCUMENT-IDENTIFIER: JP 04257034 A

TITLE: LOGICAL NAME DISPLAY SYSTEM FOR MACRO FILE

PUBN-DATE: September 11, 1992

INVENTOR-INFORMATION:

NAME
OKURA, TAKAO

COUNTRY

INT-CL (IPC): G06F 11/28; G06F 12/00

Full	Title	Citation	Front	Review	Classification	Date	Reference	Sequences	Attachments	Claims	KWIC	Draw Desc	Clip Img	Image
------	-------	----------	-------	--------	----------------	------	-----------	-----------	-------------	--------	------	-----------	----------	-------

☐ 4. Document ID: CN 1361891 A WO 200023919 A1 AU 200013152 A EP 1121654 A1
BR 9914551 A JP 2002528794 W

L1: Entry 4 of 5

File: DWPI

Jul 31, 2002

DERWENT-ACC-NO: 2000-364943

DERWENT-WEEK: 200279

COPYRIGHT 2003 DERWENT INFORMATION LTD

TITLE: Extensible macro language providing method for use in computer language processors, involves retrieving code associated with keywords representing new macro command, which is then executed

INVENTOR: DEFFLER, T A; MINTZ, E

PRIORITY-DATA: 1998US-104682P (October 16, 1998)

PATENT-FAMILY:

PUB-NO	PUB-DATE	LANGUAGE	PAGES	MAIN-IPC
CN 1361891 A	July 31, 2002		000	G06F017/30
WO 200023919 A1	April 27, 2000	E	031	G06F017/30
AU 200013152 A	May 8, 2000		000	G06F017/30
EP 1121654 A1	August 8, 2001	E	000	G06F017/30
BR 9914551 A	March 5, 2002		000	G06F017/30
JP 2002528794 W	September 3, 2002		020	G06F009/45

INT-CL (IPC): G06 F 9/45; G06 F 17/30

Full	Title	Citation	Front	Review	Classification	Date	Reference	Sequences	Attachments
------	-------	----------	-------	--------	----------------	------	-----------	-----------	-------------

KWIC	Draw Desc	Clip Img	Image
------	-----------	----------	-------

☐ 5. Document ID: SU 1247882 A

L1: Entry 5 of 5

File: DWPI

Jul 30, 1986

DERWENT-ACC-NO: 1987-085544

DERWENT-WEEK: 198712

COPYRIGHT 2003 DERWENT INFORMATION LTD

TITLE: Microcommands memory two-level control - has extended logic handling of micro command details offering scope for modification to whole or parts

INVENTOR: MELNIKOV, V A

PRIORITY-DATA: 1984SU-3820013 (December 3, 1984)

PATENT-FAMILY:

PUB-NO	PUB-DATE	LANGUAGE	PAGES	MAIN-IPC
SU 1247882 A	July 30, 1986		004	

INT-CL (IPC): G06F 12/00

Full	Title	Citation	Front	Review	Classification	Date	Reference	Sequences	Attachments
------	-------	----------	-------	--------	----------------	------	-----------	-----------	-------------

KWIC	Draw Desc	Image
------	-----------	-------

  Generate Collection

Print

Terms	Documents
extend\$ near3 macro with command	5

Display Format:

-

Change Format

[Previous Page](#)[Next Page](#)

WEST

Generate Collection

Print

L16: Entry 90 of 298

File: USPT

Sep 17, 2002

DOCUMENT-IDENTIFIER: US 6453356 B1
TITLE: Data exchange system and method

Detailed Description Text (113):

Two macros, DX_SYSINIT and DX_SYSEXIT, are used to manage initialization and destruction of the DX_SysConfigObject, respectively. A usage example of these two macros is given as follows:

Detailed Description Text (125):

In order to write a message into the log/trace file, the developer may use the macro DX_TL as shown below: DX_TL(DX_ARGS,Category, StringToLog/ErrorNumber[,arg 1 [,arg2]]);

Detailed Description Text (126):

The macro DX_ARGS includes parameters such as filename, line number, time and thread ID that are automatically written into the trace/log messages. Category is specified by the following enumerated data types:

Detailed Description Text (134):

An error/event may occur at a very low level in the code (e.g., database space exhausted). It is important to report this low level event, but it is also important to report the context of what was trying to be achieved within the application when this low level error occurred. The application developer is provided with macros to define a context within the developer's code. The set of macros provided for this purpose include: INIT_CONTEXT; CONTEXT_BEGIN; and CONTEXT_END. In general, every function using the context macros should first use the macro INIT_CONTEXT. It is noted that, if INIT_CONTEXT is not called before defining CONTEXT_BEGIN, the code may not compile.

Detailed Description Text (135):

The beginning of a context may be defined using the macro CONTEXT_BEGIN, and the end of a context can be defined using the macro CONTEXT_END, as is indicated in the following example. The CONTEXT_BEGIN macro takes the argument Context Number. This context number is used to access the Context Catalog of an application and to retrieve the context string. It is noted that nested contexts are generally not allowed. If a CONTEXT_BEGIN is called before the previous context is ended, an implicit CONTEXT_END for the previous context is assumed. The following example is provided:

Detailed Description Text (140):

Within a given function, INIT_CONTEXT declares a pointer to a DX_ContextObject, referred to as dx_context, and initializes it to point to a global dummy DX_ContextObject, whose context string is blank. It also declares and initializes a variable dx_init_context. The definition of the INIT_CONTEXT macro is as follows:

Detailed Description Text (142):

The macro CONTEXT_BEGIN, described in the following example, checks whether dx_init_context is initialized or not. The significance of this check is to make sure that the function does not compile if INIT_CONTEXT is not called before the first occurrence of CONTEXT_BEGIN. It then initializes the DX_ContextObject pointer to point to a new DX_ContextObject instance storing the context string specified by the context number argument.

Detailed Description Text (144):

The macro CONTEXT_END deletes the DX_ContextObject instance created by CONTEXT_BEGIN, as can be seen in the following example.

Detailed Description Text (169):

As in the case of run-time configuration management, the ReconfigParameters()

function on `DX_SysConfig` object will be called. In this function, the `DX_SysConfigObject` first checks if the signal/event received corresponds to Shutdown and if the PID specified is its own PID. If so, it, in turn, must make sure that no new transactions are started, and waits for all of the current transactions to be completed. This involves calling the macro `DX_SYSEXIT`. It is noted that, before shutting down, the entry in the configuration file should be deleted by the exiting process. It is possible that the component aborts prior to cleaning up the configuration file. This stray entry does not effect the start up of any other component using the same configuration file. `DX_ConfigSet` is also responsible for clean up of stray `DX_SHUTDOWN` entries in the configuration file.

Detailed Description Text (180):

The `DX_ThreadController` is implemented as a singleton object which makes system tuning and performance management an easier task. The `DX_ThreadController` is instantiated by the `DX_SysConfigObject` at startup. All parameters used by the `DX_ThreadController` are configurable at runtime via the `DX_ConfigSet` tool, with any changes being applied to the next thread created following the configuration change.

Detailed Description Text (181):

A macro called `DX_Thread_Execute()` is provided for ease of use. This macro retrieves the `DX_ThreadController` instance from the `DX_SysConfigObject` and then invokes the `DX_ThreadController::Execute()` method. The method `DX_ThreadController::Execute()` behaves exactly the same as if a call was invoked to create a new thread. A pointer must be passed to the function and as well as a pointer to the arguments. Internally, the `DX_ThreadController` uses the class `DX_ThreadRequest` when a thread is not available to provide a FIFO buffer that will store the function pointer and argument pointer. Each time a thread completes execution, the FIFO is checked for the presence of entries. If there are entries in the FIFO, the first entry in the buffer is removed and executed. An example of `DX_ThreadController` implementation is provided in the following example:

WEST[Help](#)[Logout](#)[Interrupt](#)[Main Menu](#)[Search Form](#)[Posting Counts](#)[Show S Numbers](#)[Edit S Numbers](#)[Preferences](#)[Cases](#)**Search Results -**

Terms	Documents
macro and uncompile	0

Database:

US Patents Full-Text Database
US Pre-Grant Publication Full-Text Database
JPO Abstracts Database
EPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Search:[Refine Search](#)[Recall Text](#)[Clear](#)**Search History****DATE:** Wednesday, May 21, 2003 [Printable Copy](#) [Create Case](#)

Set Name **Query**
side by side

Hit Count **Set Name**
result set

DB=USPT,PGPB,JPAB,EPAB,DWPI,TDBD; PLUR=YES; OP=OR

<u>L17</u>	macro and uncompile	0	<u>L17</u>
<u>L16</u>	L15 and runtime	298	<u>L16</u>
<u>L15</u>	macro and compile	1131	<u>L15</u>
<u>L14</u>	L13 and runtime	36	<u>L14</u>
<u>L13</u>	maco and compile	54	<u>L13</u>
<u>L12</u>	L11 and runtime	9	<u>L12</u>
<u>L11</u>	marco and compile	34	<u>L11</u>
<u>L10</u>	L9 and extend\$ near3 macro	0	<u>L10</u>
<u>L9</u>	marco and (runtime or "run time")	86	<u>L9</u>
<u>L8</u>	l1 and (runtime or "run time")	0	<u>L8</u>
<u>L7</u>	l1 and compil\$	2	<u>L7</u>
<u>L6</u>	l1 and recompil\$	0	<u>L6</u>
<u>L5</u>	l1 and "without recompil\$"	0	<u>L5</u>
<u>L4</u>	l1 and without recompil\$	2613	<u>L4</u>
<u>L3</u>	"extended macro command"	0	<u>L3</u>
<u>L2</u>	extend\$ near3 macro near3 command	2	<u>L2</u>
<u>L1</u>	extend\$ near3 macro with command	5	<u>L1</u>

END OF SEARCH HISTORY

WEST[Generate Collection](#)[Print](#)**Search Results - Record(s) 1 through 1 of 1 returned.**☐ 1. Document ID: US 5768593 A

L48: Entry 1 of 1

File: USPT

Jun 16, 1998

US-PAT-NO: 5768593

DOCUMENT-IDENTIFIER: US 5768593 A

TITLE: Dynamic cross-compilation system and method

DATE-ISSUED: June 16, 1998

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Walters; Chad Perry	Redwood City	CA		
Brown; Jorg Anthony	Concord	CA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Connectix Corporation	San Mateo	CA			02

APPL-NO: 08/ 620387

DATE FILED: March 22, 1996

INT-CL: [06] G06 F 9/30, G06 F 9/44

US-CL-ISSUED: 395/705; 395/707, 395/709, 395/500, 395/581

US-CL-CURRENT: 717/141; 703/26, 712/234

FIELD-OF-SEARCH: 395/700, 395/701-710, 395/712, 395/500, 395/581

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
<u>4667290</u>	May 1987	Goss et al.	395/707
<u>5167023</u>	November 1992	De Nicolas et al.	395/527
<u>5179703</u>	January 1993	Evans	395/703
<u>5204960</u>	April 1993	Smith et al.	395/707
<u>5367683</u>	November 1994	Brett	395/709
<u>5625822</u>	April 1997	Brett	395/705
<u>5649203</u>	July 1997	Sites	395/709
<u>5692196</u>	November 1997	Unni et al.	395/705

ART-UNIT: 274

PRIMARY-EXAMINER: Voeltz; Emanuel Todd

ASSISTANT-EXAMINER: Dam; Tuan Q.

ABSTRACT:

In a computer system, a cross-compiler converts non-native code into native code immediately prior to execution of that code. The system also includes a code cache for storing cross-compiled code and a hash table for locating code blocks in the code cache. In a preferred embodiment, the system also includes an interpreter for emulating certain non-native instructions that are not converted into native code by the cross-compiler. While executing any non-native application, if the next instruction is not one of the predefined set of non-native instructions to be handled by interpretation or a special purpose procedure, then the next instruction is considered to be an "entry point" instruction, and the cross-compiler looks up the address of the entry point instruction in the hash table to see if a corresponding native code block is already stored in the code cache. If so, the native code block in the code cache is executed until an exit instruction in the native code block is encountered. Otherwise, the cross-compiler cross-compiles all code that is reachable from the entry point instruction during execution of the program without going outside the compilation window. During compilation the cross-compiler determines the non-native condition codes generated by a non-native instruction that will not be used by any successors of the non-native instruction. The native code instructions generated by the cross-compiler do not include instructions for processing non-native condition codes generated by the non-native instruction that will not be used by any successors of the qualifying non-native instruction.

12 Claims, 6 Drawing figures

Full	Title	Citation	Front	Review	Classification	Date	Reference	Sequences	Attachments	Claims	KWIC	Draw Desc	Image
------	-------	----------	-------	--------	----------------	------	-----------	-----------	-------------	--------	------	-----------	-------

[Generate Collection](#)[Print](#)

Terms	Documents
5768593.pn.	1

Display Format:

TI

[Change Format](#)[Previous Page](#)[Next Page](#)

WEST

Generate Collection

Print

Search Results - Record(s) 1 through 1 of 1 returned.☐ 1. Document ID: US 6151703 A

L45: Entry 1 of 1

File: USPT

Nov 21, 2000

US-PAT-NO: 6151703

DOCUMENT-IDENTIFIER: US 6151703 A

TITLE: Development system with methods for just-in-time compilation of programs

DATE-ISSUED: November 21, 2000

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Crelrier; Regis	Santa Cruz	CA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Inprise Corporation	Scotts Valley	CA			02

APPL-NO: 08/ 650512

DATE FILED: May 20, 1996

INT-CL: [07] G06 F 9/44

US-CL-ISSUED: 717/5; 717/6, 717/7

US-CL-CURRENT: 717/136

FIELD-OF-SEARCH: 395/705, 717/5, 717/6, 717/7

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
<u>5768593</u>	June 1998	Walters	395/705

OTHER PUBLICATIONS

Gosling, J., and McGilton, H., The Java Environment: A White Paper, Sun Microsystems, Inc., Oct. 1995.

JAVA: The First 800 Days, Sun Microsystems Inc website

<http://java.sun.com/events/jibe/timeline.html>, Jun. 3, 2000.

JAVA On Solaris 2.6 A White Paper, Sun Microsystems Inc., 1997.

JAVA JIT Compiler Overview Sun Microsystems Inc. <http://www.sun.com/solaris/jit>, Jun. 21, 2000.

Design and Implementation of Pep, a JAVA Just-In-Time Translator, O Agesen, Theory and Practice of Object Systems v2, No. 2 pp. 127-155, 1997.

"Vlatte: A JAVA VM Just-In-Time Scheduling Compiler" Mass Laboratory, Seoul National University, <http://latte.snu.ac.kr/vlatte>, Jun. 20, 2000.

Latte: A Fast and Efficient Java VM Just-in-Time Compiler, Mass Laboratory, Seoul National University, <http://latte.snu.ac.kr>, Jun. 20, 2000.

Welcome to Micro-Architecture and System Software Laboratory Mass Laboratory, Seoul National University, <http://altair.snu.ac.kr/>, Jun. 20, 2000.

Electrical Fire A Compiler for the JAVA platform, mozilla.org
wysiwyg://26/http://www.mozilla.org/projects/ef/, Jun. 20, 2000.
"Java Code Brews For Embedded Apps", Sreeram Duvuurrur Electronic Engineering Times n
932, p. 70, Dec. 16, 1996.
Borland Announces Availability of Borland CH Development Suite 5.0 Business Wire,
Mar.26, 1996.
JAVA Unleashed, Michael Morrison et al. Macmillan Computer Pub, ISBN 1575210495, Apr.
12, 1996.
SPICE: A System for Translating Smalltalk Programs Into a C Environment, IEEE
Transactions on Software Engineering V21 No. 11, Nov. 1996.
Brihi: an Optimizing Java Compiler, M. Cierniak et al, University of Rochester,
IEEE, 1997.
Object Share Company Press Release, Visual Works 5i [http://www/objectshare.com](http://www.objectshare.com), Jun.
19, 2000.
Compiling JAVA Just In Time, IEEE Micro pp. 36-43, T. Cramer et al., 1997.
The JAVA Hotspot Performance Engine Architecture p. 1-13
<http://java.sun.com/products/hotspot/whitepaper.html>, Apr. 1999.
CS265 "Expert " p.: Just In Time Compilers, Matt Welsh, UC Berkeley,
<http://www.cs.berkeley.edu/.about.mdw/class/cs265/>, Feb. 14, 2000.
Dynamic Compilation, Westley Weiner: CS 265 Topic UC Berkeley,
<http://www.cs.berkeley.edu/.about.weiner/cs265.html>, Jun. 21, 2000.
Not Just Making Smalltalk--Parcplare will move beyond signature products with plug
in support for Alternative Technology p. 83 Computer Tech., May 27, 1996.
Parc Place Throws In Smalltalk Towel; Heads For Java, Network Briefing, Jul. 23,
1997.
Hot Spot At Center of Sun Java Plans, Electronic News Jim De Tar, Sep. 1, 1997.
Analysis and Compilation of Object Oriented Languages,
<http://www.csd.uv.se/.about.thomas/wpo/oo-compilation-papers.html>, Jun. 21, 2000.
Feedback--Directed Compilation
<http://www.cs.berkeley.edu/.about.richie/cs265/feedback/>, Jun. 21, 2000.
Design, Implementation, and Evaluation of Optimizations in a Just-In-Time Compiler,
K. Ishizaki et al, No Date.

ART-UNIT: 272

PRIMARY-EXAMINER: Hafiz; Tariq R.

ASSISTANT-EXAMINER: Ingberg; Todd

ABSTRACT:

A development system having a client which employs a virtual machine for executing programs written in the Java programming language is described. The client executes a "compiled" (i.e., bytecode or pseudo-compiled) Java program, which has been created by compiling a Java source code program or script with a Java compiler. The pseudo-compiled program comprises the bytecode emitted by the compiler. The development system further includes a just-in-time compiler which natively compiles each pseudo-compiled method of a Java program on a "just-in-time" basis--that is, compiles each method as it is actually used into native machine code for a target microprocessor. Methods which are unused are left uncompiled (i.e., left as bytecode). During program execution, when a method call is made from interpreted code, the system employs an "invoker" slot of the callee. When a method call is made from compiled code, the system employs a "compiled code" slot of the callee. As the addresses for the slots themselves remain unchanged, a method which has been compiled need not be recompiled when a callee method it invokes is itself compiled. In this manner, a method (caller) calling another method (callee) need not know whether it is calling is an interpreted method or a compiled method.

23 Claims, 8 Drawing figures

Full Title CLS:1 REF:1 SEQ:1 ATT:1

Generate Collection

Print

WEST☐ **Generate Collection** **Print**

L35: Entry 4 of 8

File: USPT

Aug 13, 1996

US-PAT-NO: 5546583

DOCUMENT-IDENTIFIER: US 5546583 A

TITLE: Method and system for providing a client/server interface in a programming language

DATE-ISSUED: August 13, 1996

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Shriver; David I.	Euleess	TX		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE	CODE
International Business Machines Corporation	Armonk	NY			02	

APPL-NO: 08/ 223276 [PALM]

DATE FILED: April 5, 1994

INT-CL: [06] G06 F 13/00

US-CL-ISSUED: 395/650; 364/DIG.1, 364/284.4

US-CL-CURRENT: 709/313; 709/330

FIELD-OF-SEARCH: 395/650

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

☐ Search Selected☐ Search ALL

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
<input type="checkbox"/>	<u>5086504</u>	February 1992	Nemeth-Johannes et al.	395/700
<input type="checkbox"/>	<u>5255386</u>	October 1993	Prager	395/600
<input type="checkbox"/>	<u>5257366</u>	October 1993	Adair et al.	395/600
<input type="checkbox"/>	<u>5287514</u>	February 1994	Gram	395/700
<input type="checkbox"/>	<u>5291585</u>	March 1994	Sato et al.	395/500
<input type="checkbox"/>	<u>5317722</u>	May 1994	Evans	395/500
<input type="checkbox"/>	<u>5430876</u>	July 1995	Schreiber et al.	395/650

OTHER PUBLICATIONS

Shriver, David I., "REXX in the CICS Environment", Third REXX Symposium Annapolis, Maryland, 1992, pp. 1-41.
Shriver, David I., "Research on REXX in the CICS Environment", Share 80 San Francisco I916, 1993, pp. 1-44.

Shriver, David I., "Research on REXX in the CICS Environment", Share 77 Chicago, Illinois 1940, 1991, pp. 1-36.

ART-UNIT: 236

PRIMARY-EXAMINER: Heckler; Thomas M.

ABSTRACT:

In a data processing system, a programming language processor capable of executing program code is provided. A client program and a server program are also provided within said data processing system. The client program and the server program are comprised of program code capable of execution within said data processing system. Once the client and server programs are invoked, the client program sends a request for a service to the server program. In response to program code within the server program, a request is sent to the client program for a service that requires access to a variable within the client program. The client program then processes the request from the server program and sends the server program a response. Thereafter, the server program continues processing the request from the client program in response to gaining access to the variable in the client program. If the server program has not been initialized when the client program requests a service, the client program automatically initializes the server program.

8 Claims, 6 Drawing figures

WEST

Generate Collection

Print

L35: Entry 4 of 8

File: USPT

Aug 13, 1996

DOCUMENT-IDENTIFIER: US 5546583 A

TITLE: Method and system for providing a client/server interface in a programming language

Brief Summary Text (15):

In the role of a macro language, the procedures language interfaces with the command line interface presented by an application to the application user. For example, the procedures language may interface with a text editor, such as XEDIT on VM or KEDIT on the PC. This allows the application user to utilize the procedures language to personalize the application by grouping together application commands in conjunction with procedures language logic and, if needed, system commands. Users may utilize a sequence of commands presented by the procedures language to the application to perform repetitive tasks, and extend the application user interface.

Brief Summary Text (18):

Instructions are identified by a REXX keyword or a group of REXX keywords specifying a particular task.

Brief Summary Text (22):

During execution, three types of clauses require action: (1) instructions which are recognized as REXX keywords are executed; (2) assignments are made; and (3) commands are executed. In such execution of system commands, strings that are not recognized as null clauses, labels, assignments, or instructions, are passed to the calling environment for execution.

WEST☐

L35: Entry 2 of 8

File: USPT

Feb 29, 2000

US-PAT-NO: 6031993

DOCUMENT-IDENTIFIER: US 6031993 A

TITLE: Method and apparatus for translating source code from one high-level computer language to another

DATE-ISSUED: February 29, 2000

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Andrews; Kristy A.	Palo Alto	CA		
Del Vigna; Paul	San Jose	CA		
Molloy; Mark E.	San Jose	CA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Tandem Computers Incorporated	Cupertino	CA			02

APPL-NO: 09/ 006138 [PALM]

DATE FILED: January 13, 1998

PARENT-CASE:

This application is a continuation and claims the benefit of U.S. application Ser. No. 08/319,682, filed Oct. 7, 1994, now U.S. Pat. No. 5,768,564 the disclosure of which is incorporated by reference.

INT-CL: [07] G06 F 9/45

US-CL-ISSUED: 395/707; 395/708, 707/100

US-CL-CURRENT: 717/143; 707/100, 717/136, 717/144, 717/146

FIELD-OF-SEARCH: 395/705, 395/707, 395/701-703, 395/708, 707/100, 707/101, 707/102, 707/104

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
<input type="checkbox"/>	<u>5477451</u>	December 1995	Brown et al.	395/500
<input type="checkbox"/>	<u>5852740</u>	December 1998	Estes	395/800.15

OTHER PUBLICATIONS

Merlo et al., "Structural and behavioral code representation for program understanding", Proc. of CASE Workshop, IEEE, 1992, pp. 106-108.
Hatcher et al., "A production quality C compiler for Hypercube Multiprocessors" ACM SIGPLAN Notices, vol. 26, No. 7, Jul. 1991, pp. 73-82.

Heun, Optimal dynamic edge-disjoint embeddings of complete binary trees into hypercubes, Compuscience, p. 14, Jan. 1, 1996.

ART-UNIT: 272

PRIMARY-EXAMINER: Hafiz; Tariq R.

ASSISTANT-EXAMINER: Chaki; Kakali

ABSTRACT:

A method, system, apparatus, and program for translating one computer language to another using doubly-rooted tree data structures. A doubly-rooted tree is the combination of two sets of hierarchically related objects sharing a common set of leaves. An N-rooted tree is also described. When a doubly-rooted tree is constructed in the specified manner and then translated to a second doubly-rooted tree, source language code is transformed into target language code. In addition, the translation preserves preprocessor characteristics of the source language code including macros, conditionally compiled regions of code, source inclusion statements, and comments.

6 Claims, 12 Drawing figures

WEST☐ **Generate Collection** **Print**

L37: Entry 11 of 19

File: USPT

Jun 5, 1990

US-PAT-NO: 4931928

DOCUMENT-IDENTIFIER: US 4931928 A

TITLE: Apparatus for analyzing source code

DATE-ISSUED: June 5, 1990

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Greenfeld; Norton R.	Wayland	MA	01778	

APPL-NO: 07/ 269135 [PALM]

DATE FILED: November 9, 1988

INT-CL: [05] G06F 9/44

US-CL-ISSUED: 364/300

US-CL-CURRENT: 717/131; 707/3, 717/114, 717/142, 717/143

FIELD-OF-SEARCH: 364/200, 364/300, 364/900

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

☐ **Search Selected**☐ **Search ALL**

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
<input type="checkbox"/>	<u>4506326</u>	March 1985	Shaw et al.	364/300
<input type="checkbox"/>	<u>4686623</u>	August 1987	Wallace	364/300
<input type="checkbox"/>	<u>4688195</u>	August 1987	Thompson et al.	364/300
<input type="checkbox"/>	<u>4729096</u>	March 1988	Larson	364/300
<input type="checkbox"/>	<u>4751635</u>	June 1988	Kret	364/200
<input type="checkbox"/>	<u>4787035</u>	November 1988	Bourne	364/300

OTHER PUBLICATIONS

"Global Program Analysis In An Interactive Environment" by Larry M. Masinter, SSL-80-1 Xerox Palo Alto Research Center, Palo Alto, Calif., Jan. 1980, pp. 39-61 and 67-83 (Chapter 4-6 and Appendixes 1-3 respectively).

"Telescope: A Cross Reference Utility for Lisp" by Jed Krohnfeldt, Utah PASS Project Op Note 86-11, Dec. 4, 1986, pp. 1-2.

"Reverserver: Databases for Reverse Engineering" in Release 1.0, published by EDventure Holding, Inc., Apr. 10, 1989, pp. 12-13.

ART-UNIT: 232

PRIMARY-EXAMINER: Zache; Raulfe B.

ABSTRACT:

Apparatus in a computer system provides source code analysis. The apparatus includes an analysis member which extracts programming semantics information from an input source code. The analysis member operates according to the programming language of the source code as defined by a grammar mechanism. The analysis member employs a database interface which enables the extracted programming semantics information to be placed in a user desired database for subsequent recall by a desired query system. The database and query system may be pre-existing elements which are supported by a digital processor independently of the analysis member. A relational database with an SQL query system may be used.

19 Claims, 8 Drawing figures

WEST

Generate Collection

Print

L37: Entry 11 of 19

File: USPT

Jun 5, 1990

DOCUMENT-IDENTIFIER: US 4931928 A

TITLE: Apparatus for analyzing source code

Detailed Description Text (67):

The "C" programming language also defines a preprocessor 46, which is a text-oriented macro language coexistent with "C". This preprocessor 46 is implemented as a separate program. In the preferred embodiment for the "C" target language, the preprocessor 46 is integrated with the lexical scanner 40 for efficiency. Thus the presence of preprocessor commands makes the lexical scanner 40 call a preprocessor command subprogram 64 to analyze the preprocessor line. The preprocessor command subprogram 64 contains a separate formal grammar of legal statements for use in certain expressions, and this is again processed by the LALR(1) generator 58. The preprocessor subsystem 46 keeps a separate macro symbol table 68, and the lexical scanner 40 checks the macro symbol table 68 whenever it scans a symbol. If the symbol has a macro definition that definition is processed by a preprocessor macro expand subprogram 66 at that time. For each scanned symbol, the preprocessor subprograms 64 and 66 replace the symbol by its definition and process that definition by passing the definition to the lexical scanner 40, directly or indirectly through the source file input. The preprocessor subsystem 46 also extracts semantics information as described below.

WEST☐ **Generate Collection** **Print**

L37: Entry 6 of 19

File: USPT

Sep 22, 1998

US-PAT-NO: 5812853

DOCUMENT-IDENTIFIER: US 5812853 A

TITLE: Method and apparatus for parsing source code using prefix analysis

DATE-ISSUED: September 22, 1998

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Carroll; Martin D.	Watchung	NJ		
Juhl; Peter	Vestbjerg			DK
Koenig; Andrew Richard	Gillette	NJ		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Lucent Technologies Inc.	Murray Hill	NJ			02

APPL-NO: 08/ 225880 [PALM]

DATE FILED: April 11, 1994

INT-CL: [06] G06 F 9/45

US-CL-ISSUED: 395/708; 395/707

US-CL-CURRENT: 717/143; 717/108, 717/116

FIELD-OF-SEARCH: 395/700, 395/705, 395/707, 395/701, 364/280, 364/280.4, 364/280.5

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

☐ **Search Selected**☐ **Search ALL**

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
<input type="checkbox"/>	<u>4464650</u>	August 1984	Eastman et al.	341/51
<input type="checkbox"/>	<u>4667290</u>	May 1987	Goss et al.	364/300
<input type="checkbox"/>	<u>5276880</u>	January 1994	Platoff et al.	395/700
<input type="checkbox"/>	<u>5313387</u>	May 1994	McKeeman et al.	364/400
<input type="checkbox"/>	<u>5325531</u>	June 1994	McKeeman et al.	395/700
<input type="checkbox"/>	<u>5355493</u>	October 1994	Silberbauer et al.	395/701
<input type="checkbox"/>	<u>5386570</u>	January 1995	Lindhorst	395/707
<input type="checkbox"/>	<u>5408603</u>	April 1995	Van De Lavoie et al.	395/161

OTHER PUBLICATIONS

Aho et al., "Compilers Principles, Techniques, and Tools," Addison-Wesley Publishing, Reading, MA, sect's 1.1, 1.2, and 2.4 and pp. 100, 114, 216, and 294,

1988.

Ellis, Margaret A. & Stroustrup, Bjarne, The Annotated C++ Reference Manual, Addison-Wesley Publishing Company, AT&T Bell Laboratories, 1990, pp. 26-27.

Franklin, Dan & Legget, Bill, "Lucid Energize Programming System for Sun SPARC," C++ Report, Jul./Aug. 1993, pp. 60-63, 65-66.

Murray, Robert B., "A Statically Typed Abstract Representation for C++ Programs," C++ Technical Conference, 1992, Usenix Association, pp. 83-97.

Borland C++ Version 3.0: User's Guide, Appendix D, pp. 195-199.

ObjectCenter Reference Version 2, CenterLine Software, Inc., pp. 241-245.

Symantec C++ For Windows and DOS: Compiler and Tools Guide, Symantec Corporation Corporation, 1993, pp. 56-61.

Microsoft Visual C++ Development System for Windows Version 1.0: Professional Tools User's Guides, Microsoft Corporation, pp. 88-93.

ART-UNIT: 274

PRIMARY-EXAMINER: Voeltz; Emanuel Todd

ASSISTANT-EXAMINER: Corcoran, III; Peter J.

ABSTRACT:

A method and apparatus for processing source code in a language processing system with improved parsing based on prefix analysis. A method in accordance with the present invention includes the steps of identifying a previously-parsed prefix of a source code translation unit; creating a parser in a parser state corresponding to the identified prefix; and parsing a remaining portion of the translation unit after the prefix using the parser in the parser state corresponding to the prefix. In one embodiment of the invention, the step of creating a parser includes retrieving stored level-one subtrees corresponding to the top-level statements in the prefix. The level-one subtrees corresponding to the prefix may be stored in the form of a prefix tree along with the text of the top-level source code statements represented by the prefix and a parser delta indicating the effect of the code statements on the parser state.

25 Claims, 9 Drawing figures

WEST

Generate Collection

Print

L37: Entry 6 of 19

File: USPT

Sep 22, 1998

DOCUMENT-IDENTIFIER: US 5812853 A

TITLE: Method and apparatus for parsing source code using prefix analysis

Detailed Description Text (5):

The exemplary compiler 20 includes a preprocessor 36. The preprocessor 36 generally modifies the source code in accordance with a given set of preprocessor options, and other instructions, also referred to as preprocessor directives, which are contained in the source code. In the C and C++ programming languages, for example, the preprocessor directive "#include" directs the preprocessor to read a specified file and insert it in the source code at the location of the directive. The preprocessor also expands macros into source code statements. In the C and C++ programming languages, macros may be created using the "#define" preprocessor directive. During macro-expansion, the preprocessor replaces any occurrence of a defined macro with its corresponding source code statement or statements. These and other functions of preprocessor 36 are well-known in the art and will typically vary depending upon the programming language. It should be noted that the term "macro-expanded" is also used herein to refer to source code which has been preprocessed and the term should not be construed as limiting preprocessing to only the expansion of macros.

WEST☐ **Generate Collection** **Print**

L37: Entry 4 of 19

File: USPT

Jul 4, 2000

US-PAT-NO: 6085120

DOCUMENT-IDENTIFIER: US 6085120 A

TITLE: Data system processing and method for creating application extension

DATE-ISSUED: July 4, 2000

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Schwerdtfeger; Richard Scott	Round Rock	TX		
Thatcher; James Winthrop	Austin	TX		
Weiss; Lawrence Frank	Round Rock	TX		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE	CODE
International Business Machines Corporation	Armonk	NY			02	

APPL-NO: 08/ 971256 [PALM]

DATE FILED: November 17, 1997

INT-CL: [07] G06 F 17/00

US-CL-ISSUED: 700/90; 713/100

US-CL-CURRENT: 700/90; 713/100

FIELD-OF-SEARCH: 700/86, 700/87, 700/90, 713/100, 713/1, 713/2

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected**Search ALL**

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
<input type="checkbox"/>	<u>5247678</u>	September 1993	Littleton	395/700
<input type="checkbox"/>	<u>5252951</u>	October 1993	Tannenbaum et al.	345/156
<input type="checkbox"/>	<u>5442376</u>	August 1995	Tannenbaum et al.	345/156
<input type="checkbox"/>	<u>5568487</u>	October 1996	Sitbon et al.	370/466
<input type="checkbox"/>	<u>5628005</u>	May 1997	Hurvig	707/8
<input type="checkbox"/>	<u>5819097</u>	October 1998	Brooks et al.	395/705
<input type="checkbox"/>	<u>5854750</u>	December 1998	Phillips et al.	700/216
<input type="checkbox"/>	<u>5884078</u>	March 1999	Faustini	395/701
<input type="checkbox"/>	<u>5928360</u>	July 1999	Masuoka et al.	713/2
<input type="checkbox"/>	<u>5938766</u>	August 1999	Anderson et al.	713/100

OTHER PUBLICATIONS

Andrew Wooldridge et al., Special Edition, Using JavaScript Second Edition, published by Que Corporation, copyright 1997, pp. 16, 318-345, 366-370, 384-407.

Jill Ellsworth et al., The Internet 1997 Unleashed, published by Sams.net Publishing, copyright 1997, pp. 500, 541, 633, 635, 662-682, 693-694, 756-760, 795, 800-803, 931-933.

J.M. Gill, The Design of Man-Machine Interfaces for Use by Visually Disabled People, pp. 1-7, available via the Internet at <http://www.rib.org.uk/wedo/research/sru/japan.html>, attached copy printed Jul. 29, 1997.

IBM Special Needs Home Page, available via the Internet at <http://www.austin.ibm.com/sns/index.html>, 12 pp., attached copy printed Oct. 7, 1997.

William D. Walker et al., Making the X Window System Accessible to People with Disabilities, pp. 1-10, Oct. 8, 1997.

IBM Special Needs, IBM Screen Magnifier/2, available via the Internet at <http://www.ibm.com>, pp. 1-2, attached copy printed Oct. 8, 1997.

IBM Special Needs, IBM AccessDOS, available via the Internet at <http://www.ibm.com>, pp. 1-2, attached copy printed Oct. 8, 1997.

IBM Special Needs, Braille/2 for IBM Screen Reader/2, available via the Internet at <http://www.ibm.com>, pp. 1-2, attached copy printed Oct. 8, 1997.

Earl Johnson et al., Making the X Window System More Accessible, the DACX Project, pp. 1-10, Presentation at California State University Northridge (CSUN) Conference, Mar. 18, 1994.

Speech Viewer III Announcement, IBM Announces Speechviewer III for Windows for Interactive Speech Therapy, available via the Internet at <http://www.ibm.com>, pp. 1-2, attached copy printed Oct. 8, 1997.

IBM Special Needs, IBM Screen Readers, available via the Internet at <http://www.ibm.com>, pp. 1-2, attached copy printed Oct. 8, 1997.

IBM Special Needs, IBM Screen Magnifier/2 Update, available via the Internet at <http://www.ibm.com>, pp. 1-2, attached copy printed Oct. 8, 1997.

IBM Special Needs, Software Accessibility, available via the Internet at <http://www.ibm.com>, pp. 1-2, attached copy printed Oct. 8, 1997.

IBM Special Needs FTP Service, available via the Internet at <ftp.software.ibm.com/sns>, 1 page, attached copy printed Oct. 8, 1997.

Telecommunications and Persons with Disabilities: Building the Framework, The Second Report of The Blue Ribbon Panel on National Telecommunications Policy, pp. 1-34, attached copy printed Oct. 8, 1997.

Ellen Francik, Telephone Interfaces: Universal Design Filters, Version 2, Jun. 6, 1996, pp. 1-13, Human Factors Engineering, Pacific Bell, Jun. 6, 1996.

Apple Macintosh Software Toolkit, Mouse Cursor Enhancers (Visual), available via the Internet, pp. 1-2, attached copy printed Oct. 8, 1997.

Java Home Page, java.sun.com--The Source for Java, Sun Microsystems, Inc., Copyright 1995-97, available via the Internet, pp. 1-3, attached copy printed Oct. 9, 1997.

Gamelan, java.developer.com, Earth Web, Copyright 1997, available via the Internet, pp. 1-2, attached copy printed Oct. 9, 1997.

Java Home Page, The Java Virtual Machine Specification, available via the Internet, 1 page, attached copy printed Oct. 14, 1997.

General Input Device Emulating Interface (GIDEI) Proposal, Draft Version 2.0, Copyright 1994, pp. 1-38.

ART-UNIT: 276

PRIMARY-EXAMINER: Gordon; Paul P.

ASSISTANT-EXAMINER: Cabrera; Zoila

ABSTRACT:

A data processing system and method provide an extension to an application that is programmable and is written in the native language of the application. During operation and execution of the data processing system and method, the application extension is loaded when an associated Java Virtual Machine is initialized.

39 Claims, 4 Drawing figures

WEST**End of Result Set**☐ **Generate Collection** **Print**

L50: Entry 1 of 1

File: USPT

Jul 15, 1997

US-PAT-NO: 5649203

DOCUMENT-IDENTIFIER: US 5649203 A

TITLE: Translating, executing, and re-translating a computer program for finding and translating program code at unknown program addresses

DATE-ISSUED: July 15, 1997

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Sites; Richard Lee	Menlo Park	CA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Digital Equipment Corporation	Maynard	MA			02

APPL-NO: 08/ 580686 [PALM]

DATE FILED: December 29, 1995

PARENT-CASE:

RELATED APPLICATIONS This application is a divisional application of Richard L. Sites, U.S. application Ser. No. 07/666,196 filed Mar. 7, 1991, now U.S. Pat. No. 5,507,030, originally entitled Automatic Flowgraph Generation for Program Analysis and Translation, and as amended, entitled Successive Translation, Execution, and Interpretation of Computer Program Having Code at Unknown Locations Due to Execution Transfer Instructions Having Computed Destination Addresses. This application discloses subject matter that is related to subject matter disclosed in the following applications, assigned to Digital Equipment Corporation, the assignee of the present invention, and are incorporated by reference herein: Richard L. Sites, BRANCH RESOLUTION VIA BACKWARD SYMBOLIC EXECUTION, U.S. application Ser. No. 666,070, filed Mar. 7, 1991, issued as U.S. Pat. No. 5,428,786 on Jun. 27, 1995. Richard L. Sites, USE OF STACK DEPTH TO IDENTIFY MACHINE CODE MISTAKES, U.S. application Ser. No. 666,210, filed Mar. 7, 1991, issued as U.S. Pat. No. 5,450,575 on Sep. 17, 1995. Scott Robinson, Richard L. Sites, and Richard Witek, IMPROVED SYSTEM AND METHOD FOR PRESERVING INSTRUCTION STATE-ATOMICITY FOR TRANSLATED PROGRAM CODE, U.S. application Ser. No. 666,071, filed Mar. 7, 1991; Richard L. Sites, CROSS-IMAGE REFERENCING OF PROGRAM CODE, U.S. application Ser. No. 666,223, filed Mar. 7, 1991, issued as U.S. Pat. No. 5,317,740 on May 5, 1994; Scott Robinson and Richard L. Sites, IMPROVED SYSTEM AND METHOD FOR PRESERVING INSTRUCTION GRANULARITY FOR TRANSLATED PROGRAM CODE, U.S. application Ser. No. 666,025, filed Mar. 7, 1991, issued as U.S. Pat. No. 5,307,504 on Apr. 26, 1994; Thomas R. Benson, USE OF STACK DEPTH TO IDENTIFY ARCHITECTURE AND CALLING STANDARD DEPENDENCIES IN MACHINE CODE, U.S. application Ser. No. 666,083, filed Mar. 7, 1991, issued as U.S. Pat. No. 5,301,325 on Apr. 5, 1994; Thomas R. Benson, REGISTER USAGE TRACKING TO SUPPORT COMPILED 32-BIT CODE IN 64-BIT ENVIRONMENT, U.S. application Ser. No. 666,084, filed Mar. 7, 1991, issued as U.S. Pat. No. 5,339,238 on Aug. 16, 1994; Thomas R. Benson, MAPPING ASSEMBLY LANGUAGE ARGUMENT LIST REFERENCES ACROSS MACHINE ARCHITECTURES, U.S. application Ser. No. 666,085, filed Mar. 7, 1991, issued as U.S. Pat. No. 5,307,492 on Apr. 26, 1994; Thomas R. Benson, TRACKING VAX.TM. CONDITION CODES FOR PORTING TO RISC ARCHITECTURE, U.S. application Ser. No. 666,082, filed Mar. 7, 1991; Daniel L. Murphy, EFFICIENT AND FLEXIBLE LINK OF PROGRAM UNITS AT PROGRAM ACTIVATION, U.S. application Ser. No. 666,023, filed Mar. 7, 1991, issued as U.S. Pat. No. 5,297,291 on Mar. 22, 1994; Daniel L. Murphy, AUTOMATIC ADJUSTMENT OF

INTERFACE CONVENTIONS BETWEEN TWO DISSIMILAR PROGRAM UNITS, U.S. application Ser. No. 666,028, filed Mar. 7, 1991; Richard L. Sites, AUTOMATIC FLOWCHART GENERATION FOR PROGRAM ANALYSIS AND TRANSLATION, U.S. application Ser. No. 666,196, filed Mar. 7, 1991, issued as U.S. Pat. No. 5,507,030 on Apr. 9, 1996; Richard L. Sites, LOCATING PROGRAM CODE VIA SUCCESSIVE CODE EXECUTION AND INTERPRETATION, U.S. application Ser. No. 666,216, filed Mar. 7, 1991, issued as U.S. Pat. No. 5,287,490 on Feb. 15, 1994.

INT-CL: [06] G06 F 8/00

US-CL-ISSUED: 395/709; 395/705, 395/569, 395/500

US-CL-CURRENT: 717/156; 703/26, 712/228, 717/145, 717/159

FIELD-OF-SEARCH: 395/700, 395/375, 395/800, 395/500, 395/708, 395/709, 395/905, 395/568

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected

Search ALL

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
<input type="checkbox"/>	<u>4951195</u>	August 1990	Fogg, Jr. et al.	364/200
<input type="checkbox"/>	<u>5005119</u>	April 1991	Rumbaugh et al.	364/200
<input type="checkbox"/>	<u>5175828</u>	December 1992	Hall et al.	395/375

FOREIGN PATENT DOCUMENTS

FOREIGN-PAT-NO	PUBN-DATE	COUNTRY	US-CL
0372835	June 1990	EP	
90/01738	February 1990	WO	

OTHER PUBLICATIONS

Chow et al.; "Engineering a RISL Compiler," 1986, IEEE, New York, N.Y., pp. 132-137.
 Schinder, "Translation optimizes transfer of 8-bit programs to 16 bit," Jul. 23, 1981, Electronic Design, pp. 35-36.
 Saari, "6800 Binary Code Translator," 1987 FORML Conf. proceedings, pp 48-52.
 Bergh et al., "HP 3000 Emulation on HP Precision Architecture Computers," Hewlett-Packard Journal, Dec. 1987, pp. 87-89.
 Eve M. Tanner, "Providing Programmers with a Driver Debug Technique", Hewlett-Packard Journal, Oct. 1989, pp. 76-80.
 Program Flow Analysis: Theory and Applications, Muchnick & Jones, eds., Prentice-Hall, Englewood Cliffs, NJ, 1981, pp. 160-161, 178-179, 184-187, 264-265, 272-275, 280-283, 294-297.
 Banning, "The XDOS Binary Code Conversion System," COMPCON 89 (Sep. 27, 1989) San Francisco, CA, pp. 282-287.
 Hunter and Banning, "DOS at RISC," Byte, vol. 14, No. 12, (Nov. 1989), St. Peterborough, United States, pp. 361-368.
 Gaines, "On the Translation of Machine Language Programs," Communications of the Association for Computing Machinery, vol. 8, No. 12, (Dec. 1965), New York, NY pp. 736-741.
 S. Reiss, "PECAN: Program Development Systeem That Supports Multiple Views," IEEE Transactions on Software Engineering, SE-11, No. 3., Mar. 1985, IEEE, New York, N.Y., pp. 276-285.
 Beyond RISC!--An Essential Guide to Hewlett-Packard Precision Architecture, Wayne E. Holt, Ed., 1988, Software Research Northwest, Inc., Vashon Island, WA, pp. 225-238.
 The Handbook of Artificial Intelligence, vol. II, Barr & Feigenbaum, eds., William Kaufmann, Los Altos, CA, 1982, pp. 297-379.

ART-UNIT: 232

PRIMARY-EXAMINER: Donaghue; Larry D.

ABSTRACT:

A program is translated by automatically generating a flowgraph, using the flowgraph to analyze the program to provide information about blocks of instructions in the flowgraph, and then using the flowgraph and the information about the blocks of instructions to generate translated instructions. Due to execution transfers to computed destination addresses that are not determined prior to program execution, it is not possible to include all of the program instructions in the flowgraph. Execution transfers to these computed destinations are coded as calls to an interpreter that interprets the untranslated code. Returns are made from the interpreter to block entry points that are the first instructions in the blocks. Moreover, information about the location of untranslated instructions in an original program is discovered during execution of a partial translation of the program, and that information is used later during retranslation of the original program. This information includes origin addresses of translated instructions and corresponding destination address of untranslated instructions of execution transfers that occur during the execution of the partial translation. This feedback of information from execution to retranslation is performed after each execution of the translated program so that virtually all of the instructions in the original program will eventually be located and translated.

17 Claims, 33 Drawing figures

WEST**End of Result Set**☐ **Generate Collection** **Print**

L56: Entry 1 of 1

File: USPT

May 19, 1987

US-PAT-NO: 4667290

DOCUMENT-IDENTIFIER: US 4667290 A

TITLE: Compilers using a universal intermediate language

DATE-ISSUED: May 19, 1987

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Goss; Clinton	New York	NY		
Rosenberg; Richard	Brooklyn	NY		
Whyte; Peter	Fort Lee	NJ		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
501 Philon, Inc.	New York	NY			02

APPL-NO: 06/ 648554 [PALM]

DATE FILED: September 10, 1984

INT-CL: [04] G06F 9/44

US-CL-ISSUED: 364/300

US-CL-CURRENT: 717/147; 713/1, 717/114, 717/143

FIELD-OF-SEARCH: 364/300

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

☐ **Search Selected**☐ **Search ALL**

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
<input type="checkbox"/>	<u>4309756</u>	January 1982	Beckler	364/300
<input type="checkbox"/>	<u>4398249</u>	August 1983	Pardo et al.	364/300

OTHER PUBLICATIONS

Alfred V. Aho, Jeffrey D. Ullman, Principles of Compiler Design, 261-263, 327-349 (Third printing, Apr. 1979).

William A. Wulf, "PQCC: A Machine Relative Compiler Technology", Sep. 28, 1980.

R. Steven Glanville and Susan L. Graham, "A New Method for Code Generation", Conference Record of the Fifth Annual Symposium on Principles of Programming Languages.

K. V. Nori, U. Amman, K. Jenson, H. H. Nageli, Ch. Jacobi, "The PASCAL

Compiler: Implementtton Notes", Institut fur Informatils, Jul. 1976.

Inder-jeet S. Gujral, "Retargetable Code Generation for ADA* Compilers", Summary

prepared for submission to the 1982 SIGPLAN Symposium on Compiler Construction, Dec. 1981.

ART-UNIT: 232

PRIMARY-EXAMINER: Zache; Raulfe B.

ABSTRACT:

A method for directing a digital data processor to translate a program written in a source language into a sequence of machine executable instructions. The method consists of the translation of the source code into an intermediate language, followed by generation of object code for the target machine, the method being generally applicable to known source languages and to digital data processors.

40 Claims, 2 Drawing figures